---

title: Build a SUAPP Webapp description: Build a webapp to interact with SUAPPs. keywords: - application - build - suave - create - TypeScript - viem

---

This tutorial will show you how to build a SUAPP web application using suave-viem, our typescript SDK.

:::info

There are two different templates you can use for your SUAPP. One with minimal, TypeScript-only dependencies; and one which uses Next.

:::

eth_sign dependency

1. Confidential Compute Requests on SUAVE do not work with wallets that implement the EIP-1193 Javascript API. Therefore, we use the unsafe eth_sign method to sign CCRs, which does work, but requires that you enable this functionality in wallets like MetaMask.
    1. To do so in MetaMask, go to "Settings" -> "Advanced" -> scroll to bottom -> switch Eth_sign requests on.
2. Both templates below assume that you are running SUAVE locally.

To get a feel for how a SUAVE-enabled web app works, we've provided a couple examples. One is written in Vanilla TS, and one is written with Next.

Vanilla TypeScript Example

This project is written in vanilla TS (built with Vite), which means we directly manipulate the DOM Tree in the browser to render the site, rather than using a web framework like React/Next.

This template can be found in the suave-viem repo under examples/suave-web-demo.

Setup

Before running the example (in docker or locally), make sure you have a SUAVE devnet running locally.

You'll also need to install Foundry (forge is used to deploy contracts).

Next, clone the suave-viem repo and deploy the contract we'll be using for this demo to your local SUAVE chain:

```bash
# clone the repo

git clone https://github.com/flashbots/suave-viem.git

# deploy contracts

cd suave-viem/examples/suave ./deployContracts.sh ```
```

Run with Docker

We recommend running the Typescript examples in Docker for security, since javascript runtimes have the potential to execute arbitrary code on your machine.

If you're still in suave-viem/examples/suave/, jump back up to suave-viem/:

```bash
cd ../..
```

Now we'll build the docker image, which will build the source code and configure the example's environment to connect to your local devnet.

```bash
docker build -t suave-web-example .
```

Next, we'll run a docker container with our new image:

```bash
docker run -it suave-web-example
```

This will open a bash terminal inside docker, where you'll land in `examples/`.

To run the web example:

```
bash cd suave-web-demo bun dev
```

Now your container should be hosting the web app on http://172.17.0.2:5173. If not, look for the correct **Network** address in your terminal output.

Run Locally

:::info system dependencies

The following dependencies are required to run this example on your machine:

- foundry (forge is used to deploy contracts)
- bun (JS runtime & package manager)

:::

If you're still in `suave-viem/examples/suave/`, jump up to `suave-web-demo/`:

```
bash cd ../suave-web-demo
```

Install the project's dependencies and start the web app:

```
bash bun install bun dev
```

This template uses the same MEV-Share example contract we worked with using the Golang SDK in the previous tutorial.

> *If you're struggling with any of the above, you can also find this pure TypeScript template as a standalone repo here.*

Next.js Example

This template comes with a more extensive frontend framework, which uses Next (in TypeScript) and therefore depends on React. You can get it running by first cloning the repo and installing its dependencies.

Make sure you have previously built and symlinked suave-viem for this to work:

```
bash git clone git@github.com:andytudhope/build-a-suapp-next-ts.git cd build-a-suapp-next-ts yarn
```

Setup `forge` to compile your contracts:

```
bash cd packages/forge/ forge install forge build
```

Deploy the compiled contracts from the root directory (you need to have SUAVE running locally for this to work):

```
bash chmod +x packages/forge/deploy yarn contracts:deploy
```

You can start the frontend with:

```
bash yarn fe:dev
```

Conclusion

You now have two different templates from which to begin building your own SUAPP .

These templates demonstrate how to interact with SUAVE confidentially, both directly and with data from another domain. Follow the next tutorial to understand how.

Good luck and happy building ⚡ .